TITLE OF THE INVENTION

PROCESS FOR LEARNING THE BASIC FINITE AUTOMATON OF A PROTOCOL IMPLEMENTATION

BACKGROUND OF THE INVENTION

The present invention relates to communication protocols, and more particularly to a process for learning the basic finite automaton of a protocol implementation.

Communication partners are known to exchange data with one another in accordance with a communication protocol. An arrangement of devices for performing a data exchange in this manner is illustrated in Fig. 1. Communication partners A and B communicate with one another via a communication medium 10, such as an electrical line, by exchanging messages or protocol data units (PDUs) with one another in accordance with the communication protocol. The communication protocol constitutes a complete set of rules for the required behaviour of each of communication partners A and B. The communication partners A and B are entities in the sense of the OSI reference model which is described in detail in "ISO. Information Processing Systems - Open Systems Interconnection - Basis Reference Model" International Standard ISO/IS 7498, ISO, 1984. The appropriate and common concept for defining communication protocols is the extended finite state machine (EFSM) as described in the book by D. Hogrefe (Estelle, LOTOS und SDL: "Standard-Spezifiktionssprachen für verteilte Systeme", Springer Compass. Springer

Verlag - Berlin, Heidelberg, New York etc. 1989). It is a generalisation of the finite state machine (FSM) explained in the "Proceedings of the 1994 International Symposium on Software Testing and Analysis" (ISSTA), ACM SIGSOFT Software Engineering Notes, special issue, pages 109 to 124, August 1994.

What is required for digital communication to function smoothly is that all communication partners involved behave in conformity with common rules, the so-called protocols. Owing to the openness, diversity, complexity and bandwidth of modern communication systems, the development, standardisation and description of protocols have constantly been gaining in importance. However, error sources are proliferating at the same rate, resulting in more or less major interferences in communication systems. Among the typical error sources encountered again and again in modern broadband communication systems are:

- malfunctions or error functions due to the use of protocols or protocol variants which ought not to be used in the given context of a system installation;
- malfunctions or error functions due to configurable protocol parameters which were set to values incompatible with the given context of a system installation;
- malfunctions or error functions due to overload situations forcing individual components of the communication system to discard data.

Experience has shown that all the efforts towards ensuring the smooth functioning of communication-relevant hardware and software before their market introduction by means of suitable development processes and active test procedures is not successful in each and every case. This creates the need for troubleshooting aids which are used in a telecommunications system on the fly and are capable of detecting malfunctions or error functions, of attributing them to a system component and of providing hints on how to eliminate them.

What existing tools for on-the-fly measurements, usually referred to as protocol analyzers, have in common is that they are not capable of validating the entire communication behaviour with respect to the desired behaviour, instead requiring a considerable amount of manual testing. Consequently, there is a considerable potential here for automating rather unpleasant manual search work.

BRIEF SUMMARY OF THE INVENTION

It is therefore the object of the present invention to provide a process which allows the desired behaviour, which serves as a basis for verifying a communication, to be made available to an analyzer as fast and inexpensively as possible. In particular, this is also possible when the basic protocol does not exist in the form of a formal, machine-compatible or machine-readable specification.

In general, a test algorithm for a concrete communication protocol

cannot be developed completely automatically even if a machine-readable protocol specification is available, but rather requires a certain amount of human design work. The solution of the invention offers the advantage that, if there are not enough resources, the use of a machine-based learning process allows test algorithms to be developed almost without any human design work. In view of the very large number of protocols and protocol variants already in existence, of which not every single one is actually worth a significant amount of development work for a test machine, this is especially advantageous.

In the ever faster further development of communication systems and standards, one frequently observes a co-existence of internationally standardised protocols and proprietary developments which are to fulfil existing needs as fast as possible, before any competitors do, and even more so before the conclusion of a usually lengthy standardisation procedure. Often formal specifications for such proprietary protocol variants may only be obtained, if at all, at a high price from the manufacturer. The present invention allows the desired behaviour to be determined and made available for analysis even without any formal specification.

In a first aspect of the invention, a process for learning a basic finite automaton of a protocol implementation has the following steps: First, all the times within an example communication are categorized into equivalence classes. Subsequently, the equivalence classes are employed as states of a learned automaton. This allows the sequence of message

types to be learned, regardless of the message contents. Consequently, the states and state transitions of a finite automaton are learned.

In case an example communication has PDU (Protocol Data Unit) types, a similarity rate for each pair of times within the example communication is calculated in a particularly advantageous manner for forming the equivalence classes, the similarity rate depending on the PDU type sequence whose length is coincident for and surrounds both times. The similarity relation may be defined between two times, each within the example communication, by means of a lower bound on the value of the similarity rate such that two times fulfil the similarity relation if the similarity rate between these two times is larger than or equal to the lower bound. An equivalence relation for forming the equivalence classes may preferably be calculated by forming the transitive hull of a similarity relation between the times within the example communication.

Preferably, the PDUs of the example communication are entered as state transitions of the learned automaton, i.e., as a transition from the state whose equivalence class includes the time immediately prior to the PDU in question to the state whose equivalence class includes the time immediately after the PDU in question, marked with the PDU type in question, wherein transitions which are identical as far as starting and sequential states and PDU type are concerned are only entered once.

The above mentioned procedural steps or combinations thereof may be performed several times for overlapping partial sections of the example

communication, with the similarity relations of two overlapping partial sections each being united to form a common equivalence relation.

In order to also learn indications about the message contents in the form of context rules about message attributes, the present invention proposes a process for learning arithmetic classification rules for feature vectors from a training set having positive examples wherein first of all features derived from statistical measures are created in the form of arithmetic terms, and subsequently logic conditions are formulated on the numerical values of the features. Positive examples in this case are examples of error-free communication as opposed to examples in which protocol rules are violated (negative examples). In the present application, the training set is the example communication of a protocol machine consisting of PDUs, with the logic conditions constituting the rules for the numerical PDU field contents of a PDU sequence.

Particularly preferred is the formation of the derived features on the basis of correlation and regression coefficients on the training set for each possible feature pair, with the value of a derived feature for each training example being calculated as a sum, product, quotient or difference of two already present features or as a product of a present feature and a constant. When the conditions are being formulated, conspicuous accumulations of the values of the feature included in the training set or of the derived feature in a numerical value or within a numerical interval may be taken into consideration, the conspicuous accumulation being preferably

defined in that it maximizes the quotient of the width of the smaller one of the two gaps immediately adjacent to the numerical interval in which there are no values of the feature in question, and of the width of the largest gap within the numerical interval in which there are no values of the feature in question. Plural subclasses of the training set may be constructed by organizing the logic conditions in a disjunction of clauses, in which case one clause constitutes a conjunction of one or plural logic conditions and describes one subclass each of said training set. For characterizing the entire training set, a selection of the clauses constructed may be performed such that all elements, if possible, of the training set are selected by at least one of the clauses, and as many as possible of them by exactly one clause.

Further objects, advantages and novel features of the present invention are apparent from the following detailed description when read in conjunction with the appended claims and attached drawing.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

Fig. 1 is a block diagram view of a prior art arrangement of devices.

Fig. 2 is a block diagram view of an arrangement of devices for performing the process of the invention.

Fig. 3 is a view of the logic context of a trace analysis according to the present invention.

Fig. 4 is a view of how a PDU is composed of PDU type and message attributes according to the present invention.

Fig. 5 is a view of an example communication, restricted to the PDU types, with times defined therebetween according to the present invention.

Fig. 6 is a view of a similarity matrix established for five times of the example communication of Fig. 5 according to the present invention.

Fig. 7 is a view of a finite automaton resulting from the similarity matrix of Fig. 6 according to the present invention.

Fig. 8 is a view of a finite automaton resulting from the complete example communication of Fig. 5 according to the present invention.

Fig. 9 is a view of a minimum automaton resulting from the example communication of Fig. 5 with a maximum threshold according to the present invention.

Fig. 10 is a view of a maximum automaton resulting from the example communication of Fig. 5 with a minimum threshold according to the present invention;

Fig. 11 is a view illustrating a procedure for learning the message attributes according to the present invention.

Fig. 12 is a view of a table including features of the example communication of Fig. 11 as well as features derived therefrom according to the present invention.

Fig. 13 is a view for formulating conditions based on conspicuous

accumulations of a numerical value for a certain feature according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

As seen in Fig. 2 in which items identical with those of Fig. 1 are marked with identical reference characters, a trace analyzer 12 is coupled to the communication medium 10 so as to detect the data exchanged between the communication partners A and B. In doing so, the trace analyzer 12 reads any exchanged messages without changing them or otherwise affecting the ongoing communication between A and B. Although the trace analyzer knows the various message types of the communication, i.e., the protocol syntax, it has no knowledge of the protocol structure, i.e., how the messages are structured and how the message types are employed by the communication partners involved for performing communications. The trace analyzer 12 does not know reference protocols.

Fig. 3 shows the logic inclusion of the analyzing algorithm in the communication process. The analyzing algorithm reads the PDUs of a layer **n**, $\mathbf{n} \geq 1$, from the protocol stack and, in a manner described hereinafter, derives rules from this observation which describe the correct protocol behaviour. The prerequisite for this is that all communication participants and the lower protocol layers responsible for the transport of the layer **n** PDUs behave in accordance with the protocol. In this case,

however, the trace analyzer 12 does not receive any information on the primitives exchanged at the service access points.

Fig. 4 is an exemplary view of the structure of a PDU **a(x, y)**. Here, **a** designates the message type, while **x** and **y** relate to message attributes, i.e., data. In the illustrated example, bytes one and two specify the number 42 which indicates that **a** is of the ACK type. **x**, i.e., bytes three to six, designates the telephone number of the receiver, while **y**, i.e., bytes seven to ten, designates the telephone number of the transmitter.

Fig. 5 is a view of an example communication between entity A and entity B of Fig. 2. In this first step, first of all only the message types are of interest, whereas the message attributes are not considered for the time being. Between the message types times are defined, here times 1 to 14. A similarity matrix, see Fig. 6, is then established in such a way that at the top and on the left, the times of the example communication are entered. Between every two times within the example communication, the value is entered into the similarity matrix which is coincident with the length of the PDU type sequence that is coincident for and surrounds both times.

For the sake of clarity, Fig. 6 does not show the similarity matrix for all 14 times, but merely for the first five times. Since the respective times are directly coincident with themselves in all preceding and all subsequent times, the diagonal is filled in with "infinity". Since the similarity matrix is symmetrical to the diagonal, only the top right-hand half is dealt with in more detail hereinafter.

Consider the entry at the position first row, second column: The precursor of time 1 as shown in Fig. 5 is an **a**, while the precursor of time 2 is a **b**. The successor of time 1 is a **b**, while the successor of time 2 is an **a**. Therefore, the value to be entered into the matrix is 0.

The value in the first row, third column is obtained as follows: The precursor of time 1 is an **a** just like the precursor of time 3. What follows after time 1 just like after time 3 is the sequence **b a**. Consequently, the value to be entered into the matrix is 3. The remaining values of the similarity matrix are obtained in a similar manner.

Now, there are two approaches of how to derive equivalence classes from the similarity matrix:

In accordance with a first approach, all similarities above a certain threshold are sought for the purpose of transforming the similarity matrix into an equivalence matrix, and the relevant times are grouped together to give states of a finite automaton, until all such similarities exist between times of the same state. For example, in case of a lower threshold of 1, the times 1, 3 and 5 may be combined to give a state 1 and, as a countermove thereto, also the times 2 and 4 may be combined to give a new state 2. Then, the PDUs of the example communication are entered as state transitions of the learned automaton, i.e., as a transition each from the state whose equivalence class includes the time immediately prior to the PDU in question to the state whose equivalence class includes the time immediately after the PDU in question, marked with the relevant PDU

type. The state graph of the associated finite automaton and thus of the associated protocol is shown in Fig. 7.

In accordance with another process, the similarity matrix is multiplied by itself until a final state, the so-called equivalence matrix, is obtained, from which equivalence classes may then be derived. However, the matrix multiplication is changed to such an effect that additions are replaced with the formation of the maximum, and multiplications are replaced with the formation of the minimum of the two input values. In accordance with this approach, the entries in the first row, fifth column, and in the fifth row, first column of the similarity matrix of Fig. 6 are replaced with a 3.

In accordance with a third approach, an equivalence relation for the formation of an equivalence class is calculated by forming the transitive hull of a similarity relation between the times within the example communication.

For the purpose of determining equivalence classes, however, another approach is to determine equivalence matrices for different threshold values and then use the threshold at which the number of states is below a predetermined value.

If all 14 times of the example communication of Fig. 5 are evaluated, the result is the state graph shown in Fig. 8, i.e., the finite automaton illustrated in Fig. 8.

Figs. 9 and 10 show the two extremes for the finite automaton

which result as a function of the selection of the threshold. Fig. 9 shows the minimum automaton for a selected maximum threshold, and Fig. 10 shows the maximum automaton for a selected minimum threshold. Especially with a view to keeping computational time down, the respective procedural steps may be performed plural times for overlapping partial sections of the example communication, with the similarity relations of every two overlapping partial sections being united to form a common equivalence relation. A preferred overlapping range is 30 % - 50 % overlap.

In a second aspect of the invention, the next step is to learn the context rules for the message attributes. Irrespective of the embodiment illustrated herein, this second aspect of the invention may also be applied to the sequence of message types, without any previous first step (see above), if the associated message attributes are to be learned.

As shown in Fig. 11, the symbol sequence, i.e., the sequence of message types, which is the starting point here, is known from the previously determined finite automaton. A window is then laid over the example communication, preferably of a width of **w** = 3 to 5. For this purpose, one must keep in mind that for detecting relationships between remote PDUs, larger windows are chosen; however, the larger a window size becomes, the smaller is the number of features derivable therefrom. In the example which follows, the message attributes of the first PDU are of type **a v** and **w**, of PDU type **b x** and of the second PDU type **a y** and **z**, cf. Fig. 11.

The left half of Fig. 12 shows a compilation of the message attributes **v w x y z** as an example for the window (line 1) shown in Fig. 11, on the one hand, and for a different window (line 2), on the other hand. The right half of Fig. 12 shows features derived from the message attributes **v** to **z**, that is **y - v** and **x - z**, to be more precise. The derived features are preferably formed on the basis of correlation and regression coefficients on the example communication for each possible feature pair, wherein the value of a derived feature may be calculated as a sum, product, quotient or difference of two already present features, or as a product of a present feature and a constant. The term "present features" is intended to also cover features derived in a previous operation. The correlations to be examined are derived in an advantageous manner from common protocols.

On the basis of the derived features, a so-called **OK** criterion may then be formulated, as shown in the top right-hand portion of Fig. 11, i.e., the sequence of five message attributes **v, w, x, y, z** is **OK**, i.e., conforms to the protocol, if the following rules are fulfilled. Consequently, the **OK** criterion for the example communication **a(v,w) b(x) a(y,z)** shown in Fig. 11 is:

ok (v, w, x, y, z) : -

x = w + 1, y = v + 1, z = x

In addition to the conjunctive operation illustrated in the example, disjunctive operations may also be considered. This means that plural

subclasses of the training set may be constructed by organizing the logic conditions in a disjunction of clauses, wherein one clause constitutes a conjunction of one or plural logic condition(s) and describes a subclass each of the training set.

For formulating the conditions, conspicuous accumulations of the values of a feature present in the training set or of a derived feature in a numerical value or within a numerical interval may be taken into consideration. Fig. 13 shows the accumulation of numerical values regarding a message attribute **t**. As shown, an example communication **t** has the value 5 four times, the value 10 three times, the value 15 four times and the value 40 once. This gives rise to the assumption that there is an accumulation in the interval between 5 and 15. A conspicuous accumulation may be defined in particular in that it maximizes the quotient between the width of the smaller one of the two gaps immediately adjacent the numerical interval in which there are no values of the feature in question, and the width of the largest gap within the numerical interval in which there are no values of the numerical interval in question.

For characterizing the entire training set, a selection of the constructed clauses may be conducted such that all elements, if possible, of the training set are selected by at least one of the clauses, and as many as possible of them by exactly one clause.